E30-029CON2

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

| | | |
|---|---|---|
| In Re | : | Mathieu Gagne and Yuval Ofek |
| Serial No. | : | 10/752,256 |
| Filed | : | January 6, 2004 |
| FOR | : | METHOD AND APPARATUS FOR CASCADING DATA THROUGH REDUNDANT DATA STORAGE UNITS |
| EXAMINER | : | Mark A. Radtke |
| ART UNIT | : | 2165 |

Commissioner for Patents
P. O. Box 1450
Alexandria, VA 22313-1450

DECLARATION OF GEORGE A. HERBSTER

I George A. Herbster state that:

I am a patent attorney (Reg. No. 24,002) registered to practice in the United States Patent and Trademark Office;

I am the attorney of record in connection with the above-identified patent application.

In that capacity I held initial interviews with Messrs. Ofek and Gagne, the named inventors of the above-identified application, prepared that application and prosecuted that application. I also prepared grandparent application Serial No. 09/251,812 filed on February 17, 1999, now U. S. Patent No. 6,209,002 and the intermediate parent application Serial No. 09/740,281 filed December 19 2000, now U. S. Patent No. 6,687,718.

I am submitting this Declaration in support of the Applicants' argument that the claimed invention was made prior to January 6, 1999, the filing date of U. S. Patent No. 6,529,944 to LeCrone for a Host System for Remote Control of Mass Storage Volumes Using Cascading Commands.

During the customary and usual conduct of my practice, I make daily entries into a calendar notebook. Each entry identifies a matter I worked on with a file or docket number, a brief description of the services and the time spent. Monthly my assistant transcribes these notes from my calendar notebook into our Timeslips® time and billing program by Sage Software, Inc. All entries for the grandparent application Serial No. 09/251,812 were entered under my file E30-029.

Exhibit 1 hereto is a report generated by the Timeslips program that identifies my activities in preparing the grandparent application. I have blocked out the monetary amounts associated with each entry. As shown in Exhibit 1, there were a number of activities in connection with the above-identified pending application prior to January 6, 1999, namely:

1. A first conference with Yuval Ofek on September 18, 1998;
2. A second conference with Mathieu Gagne on October 19, 1998;

3. Additional time in November working on a draft application; and

4. The receipt of an EMC confidential document titled "Rules for TimeFinder Operations" from Mathieu Gagne. Exhibit 2 is a copy of that document dated November 12, 1998.

It has been my experience that initial conferences with inventors at EMC Corporation, the assignee of the above-identified patent application, occur only after an invention has been fully conceived.

Exhibit 3 is a letter dated December 1, 1998 sent by me to Mathieu Gagne that includes five sheets of drawings and draft claims for the above-identified patent application. As stated in the second paragraph of Exhibit 3, drawings were included with that letter. Those drawings depicted my understanding of one embodiment of the invention with three different locations and my concept of the differential split implementation of the invention. It is my recollection that these drawings were the basis for FIGS. 2 and 3 of the patent application. Based on my recollection and my usual and customary practice for preparing applications, I would send such drawings only after I felt I had a full understanding of the invention and was seeking verification that my understanding was correct.

The combination of these exhibits, my refreshed recollection and my usual and customary method of preparing

patent applications leads me to conclude that Messrs. Gagne and Ofek had a complete conception of the claimed invention at the time of my initial conferences with them. I also believe that I was diligent in coming to an understanding of the invention and the disclosed embodiment and in the preparation and filing of application Serial No. 09/251,812 on February 17, 1999.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Signature: /George A Herbster/

Typed Name: GEORGE A. HERBSTER,
Attorney, Reg. No. 24,002

EXHIBIT 1

*Law Offices of George A. Herbster*
Suite 303 Harbors Point
40 Beach Street
Manchester, MA 01944

Invoice submitted to:
EMC Corporation
35 Parkwood Drive
Hopkinton MA 01748-9103
Attn: John M. Gunther, Esq.

April 09, 2007

In Reference To: SRDF FarPoint Cascading 98-096

File: E30-029

Professional Services

| | | |
|---|---|---|
| 9/18/1998 - | GAH | Conference with Yuval Ofek re patent application |
| 10/19/1998 - | GAH | Conference with Matthew Gagne |
| 11/2/1998 - | GAH | Work re application draft |
| 11/3/1998 - | GAH | Work re application draft |
| 11/23/1998 - | GAH | Work re application draft |
| 11/24/1998 - | GAH | Work re application draft |
| 11/30/1998 - | GAH | Work re application draft |
| - | GAH | Work re application draft and telephone conference with M. Gagne and review new documents |
| 12/1/1998 - | GAH | Work re application draft |
| 12/7/1998 - | GAH | Conference with Matthew Gagne |
| 12/11/1998 - | GAH | Work re application draft -- specification |
| 12/14/1998 - | GAH | Work re application draft |
| - | GAH | Telephone conference and letter to J. Gunther |
| 12/28/1998 - | GAH | Work re application draft |
| - | GAH | Work re application draft |
| 12/30/1998 - | GAH | Conduct a search |
| 1/4/1999 - | GAH | Work re application draft |

| | | | |
|---|---|---|---|
| 1/5/1999 | - | GAH | Work re application draft and claims |
| | - | GAH | Work re claims |
| 1/14/1999 | - | GAH | Telephone conference with inventor and work re specification |
| 1/23/1999 | - | GAH | Work re revisions to specification and drawings |
| 2/9/1999 | - | GAH | Telephone conference with M. Gagne; revise application and drawings |
| 2/10/1999 | - | GAH | Telephone conference with M. Gagne re additional revisions |
| 2/11/1999 | - | GAH | Work re drawings |
| | - | GAH | Telephone conference with M. Gagne |
| 3/29/1999 | - | GAH | Preparation and forwarding of Information Disclosure Statement to USPTO |
| 4/29/1999 | - | GAH | Preparation and forwarding of missing parts to USPTO |
| 4/25/2000 | - | GAH | Work re amendment draft |
| 5/8/2000 | - | GAH | Work re amendment draft |
| 6/20/2000 | - | GAH | Work re amendment draft |
| 7/18/2000 | - | GAH | Work re amendment draft |
| 7/19/2000 | - | GAH | Work re amendment draft |
| 7/20/2000 | - | GAH | Work re amendment draft |
| 7/21/2000 | - | GAH | Review and revise amendment |
| 7/22/2000 | - | GAH | Review and forward draft to client |
| 8/1/2000 | - | GAH | Telephone conference with M. Gagne; final revisions; forward to USPTO |
| 10/31/2000 | - | GAH | Review Notice of Allowance; letter to P. Wilson |
| 11/21/2000 | - | GAH | Work re Rule 312 Amendment |
| 11/27/2000 | - | GAH | Preparation and forwarding of Issue Fee and formal drawings to USPTO |
| 1/2/2003 | - | GAH | Review office action; forward to client |

For professional services rendered

EXHIBIT 2

# Rules for TimeFinder operations

*Mathieu Gagne*
*Code 65 – 11/12/1998*
*EMC Confidential*

## I. Terminology

To describe the TimeFinder operations, we need 2 devices: one called the *standard device (STD)*, and a *BCV device (BCV)*. We use the following abbreviations to refer to particular mirror positions of these devices:

| | |
|---|---|
| SCM | Standard device's configured mirror positions |
| MMs | Moving mirror's position as a mirror of the standard device |
| DDF | Differential split tracking session, if it exists |
| MMb | Moving mirror's position as a mirror of the BCV device |
| BFM | BCV device's fixed mirrors |
| m4b | Mirror 4 (tracking mirror) of the BCV device |

We will write SCM(|) to signify *any* of the standard device's configured mirror position and SCM(&) to signify *all* of the standard device's configured mirror positions. Similarly for BFM.

We use abbreviations to describe the operations themselves. They consist of 3 letters, *xyz*, where: *x* is either *c* or *d* depending on whether the operation is *complete* or *differential*; *y* is either *d* or *r* depending on whether the operation is *direct* or *reverse*; and *z* is either *e* or *t* depending on whether the operation is *establish* or *terminate*. In terms of more familiar terminology, we thus have the following correspondence:

| | |
|---|---|
| cde | Establish |
| dde | Re-Establish |
| cre | Restore |
| dre | Incremental Restore |
| cdt | Split |
| ddt | Differential Split |
| crt | Reverse Split |
| drt | Reverse Differential Split. |

# II. Rules

We present here the rules that govern all 8 TimeFinder operations, that is, the set of axioms to which all TimeFinder behavior can be reduced. These rules *assume* the following bookkeeping is maintained for *every write IO*, at the same time a track is marked to be write pending:

1. Any write to the STD while not established is marked as invalid for MMs and marked as changed for DDF.
2. Any write to the STD while established is marked as changed for DDF.
3. Any write to the BCV is marked as invalid for m4b.
4. Any internal copy to MMs is marked for DDF (new to Code 65, introduced for the purposes of the differential split).

We identify rules new to Code 65 with either a {D} or an {R} depending on whether the rules was introduced for the purposes of the differential split or of the reverse split (of course, the reverse and/or differential split operations were introduced in Code 65). For clarity of presentation, we have divided all the rules into 4 categories, as follows.

### *A. HA Reject Conditions*

These rules form the first line of defense against possible misuse. The HA receiving the TimeFinder command executes these checks prior to dispatching the request to the DA. The checks consist mainly of verifying that (1) the source of the copy is valid (because TimeFinder operations always involve some copying) and (2) the operation of removing a mirror won't leave invalids stranded (because TimeFinder opeartions always involve moving a mirror around). Additionally, the establish-type operations (**e) check that write pendings won't get lost.

### *B. Write Pendings*

These rules indicate what happens to tracks that happen to be write pending to either the BCV or the STD device when the TimeFinder operation is invoked. There is here is fundamental difference between establish-type operations (**e) and terminate-type operations (**t): the former mostly cannot deal with write-pendings and so turn them into invalid tracks (in case they have escaped the detection of the rules in A above), while the latter are prepared to deal with them.

### *C. Insure Purpose*

These are straightforward rules that make the operation behave like it's supposed to.

### *D. Continuity/Persistence/Recovery*

These rules insure the proper working of TimeFinder through good times and bad times. *Continuity* refers to the need to manage the data through a chain of TimeFinder operations. *Persistence* means that we're not supposed to lose any data through an IML. And *recovery* covers those cases where rules have been abused (for example, the *force* bit used on a terminate operation) so that these is data loss that we need to keep track of.

## cde Establish

### *A. HA Reject Conditions*

#### 1. SCM(&) is invalid

Motivation: Need source for the copy to the BCV.
Microcode: get_invalid_counts_establish, called from verify_parms_suborder_12.

#### 2. BCV WP, if revsplitcheck bit is on {R}

Motivation: We have acknowledged the write to the host, but not destaged it. Upon the establish, we will lose the data. If the next split is a reverse split, we're in trouble.
Microcode: verify_parms_suborder_12, after vps12_cmd_ok

#### 3. BFM(|) is invalid, if revsplitcheck bit is on {R}

Motivation: If the next split is a reverse split, we do not have the source for the copy.
Microcode: get_invalid_counts_establish, called from verify_parms_suborder_12.

### *B. Write Pendings*

#### 1. Delete all write pendings to the BCV device, and mark all such as invalid for BFM(&)

Motivation: In case the next split is a differential split, we need to remember to synchronize this write. Also, in case the next split is a reverse split, in which case we will have lost data; see Ab above.
Microcode: reset_secondary_wp, called from hdc_establish.

### *C. Insure Purpose*

#### 1. Set MMs to invalid

Motivation: To copy data to the BCV.
Microcode: update_invalid_bits_for_bcv_device, called from hdc_establish.

### *D. Continuity/Persistence/Recovery*

#### 1. If BFM(|) is invalid, mark track as changed for DDF {D}

Motivation: Upon an IML, we would lose the BFM invalid mark, which probably indicates the data hasn't been synced up completely yet. In case the next split is differential, we need to remember this.
Microcode: update_invalid_bits_for_bcv_device, called from hdc_establish.

#### 2. If MMb is invalid, set MMs invalid

Motivation: Need to carry an invalid indication; should not happen because of the HA reject.
Microcode: update_invalid_bits_for_bcv_device, called from hdc_establish.

# dde Re-Establish

## *A. HA Reject Conditions*

### 1. SCM(&) is invalid

Motivation: Need source for the copy to the BCV.
Microcode: get_invalid_counts_establish, called from verify_parms_suborder_12.

### 2. BCV WP, if revsplitcheck bit is on {R}

Motivation: We have acknowledged the write to the host, but not destaged it. Upon the establish, we will lose the data. If the next split is a reverse split, we're in trouble.
Microcode: verify_parms_suborder_12, after vps12_cmd_ok

### 3. BFM(|) is invalid, if revsplitcheck bit is on {R}

Motivation: If the next split is a reverse split, we do not have the source for the copy.
Microcode: get_invalid_counts_establish, called from verify_parms_suborder_12.

## *B. Write Pendings*

### 1. Delete all write pendings to the BCV device, and mark all such as invalid for BFM(&)

Motivation: In case the next split is a differential split, we need to remember to synchronize this write. Also, in case the next split is a reverse split, in which case we will have lost data; see Ab above.
Microcode: reset_secondary_wp, called from hdc_establish.

## *C. Insure Purpose*

### 1. If m4b is invalid, make MMs invalid

Motivation: To insure a write to the BCV is over-written.
Microcode: update_invalid_bits_for_bcv_device, called from hdc_establish.

### 2. [If MMs is invalid, leave it invalid]

Motivation: Implicit rule to insure a write to the standard is carried over to the BCV.
Microcode: update_invalid_bits_for_bcv_device, called from hdc_establish.

## *D. Continuity/Persistence/Recovery*

### 1. If BFM(|) is invalid, mark track as changed for DDF {D}

Motivation: Upon an IML, we would lose the BFM invalid mark, which probably indicates the data hasn't been synced up completely yet. In case the next split is differential, we need to remember this.
Microcode: update_invalid_bits_for_bcv_device, called from hdc_establish.

### 2. If MMb is invalid, set MMs invalid

Motivation: Need to carry an invalid indication; should not happen because of the HA reject.
Microcode: update_invalid_bits_for_bcv_device, called from hdc_establish.

| cre | Restore |
|---|---|

## *A. HA Reject Conditions*

### 1. MMb is invalid

Motivation: Need source for the copy to the standard.
Microcode: get_invalid_counts_establish, called from verify_parms_suborder_12

### 2. STD is WP

Motivation: To protect the user against itself: it would not make sense to write to a device that's just about to be restored.
Microcode: establish_bcv_command, at ebc_chk_restore_wp, after lock.

### 3. BCV is WP, if revsplitcheck bit is on {R}

Motivation: We have acknowledged the write to the host, but not destaged it. Upon the restore, we will lose the data. If the next split is a reverse split, we're in trouble.
Microcode: verify_parms_suborder_12, after vps12_cmd_ok

### 4. BFM(|) is invalid, if revsplitcheck bit is on {R}

Motivation: If the next split is a reverse split, we do not have the source for the copy.
Microcode: get_invalid_counts_establish, called from verify_parms_suborder_12
Microcode: verify_parms_suborder_12, after vps12_cmd_ok

## *B. Write Pendings*

### 1. If MMb is WP, carry over cache slot to STD

Motivation: To propagate the write pending to the standard.
Microcode: pyramid: hdc_establish, at skip_reset_wp; cache: update_invalid_bits_for_bcv_device at uib_r_cyls_loop, called from hdc_estalish.

### 2. If BCV is WP, but MMb is not, delete WP, mark track not in cache and mark the slot invalid for BFM(&)

Motivation: The write pending will be propagated to the standard because of the invalidation, so we don't need to keep the WP indication; however, in case of a future reverse split, we need to mark the track invalid.
Microcode: pyramid: hdc_establish, at skip_reset_wp; cache: update_invalid_bits_for_bcv_device at uib_r_cyls_loop, called from hdc_estalish.

### 3. Delete all WP to STD

Motivation: Establish flows can not handle write pendings; this is redundant because of the HA reject, done under lock of the device.
Microcode: reset_regular_dv_wp_on_restore (same as reset_secondary_wp), called from update_-invalid_bits_for_bcv_device, called from hdc_establish.

## *C. Insure Purpose*

### 1. Make SCM(&) invalid

Motivation: To copy to the standard device.
Microcode: update_invalid_bits_for_bcv_device, called from hdc_establish.

### 2. Make MMs valid

Motivation: Insure source of copy.
Microcode: update_invalid_bits_for_bcv_device, called from hdc_establish.

## *D. Continuity/Persistence/Recovery*

### 1. If BFM(|) is invalid, mark track as changed for DDF {D}

Motivation: Upon an IML, we would lose the BFM invalid mark, which probably indicates the data hasn't been synced up completely yet. In case the next split is differential, we need to remember this.
Microcode: update_invalid_bits_for_bcv_device, called from hdc_establish.

| dre | Incremental Restore |
|---|---|

## *A. HA Reject Conditions*

### 1. MMb is invalid

Motivation: Need source for the copy to the standard.
Microcode: get_invalid_counts_establish, called from verify_parms_suborder_12

### 2. STD is WP

Motivation: To protect the user against itself: it would not make sense to write to a device that's just about to be restored.
Microcode: establish_bcv_command, at ebc_chk_restore_wp, after lock.

### 3. BCV is WP, if revsplitcheck bit is on {R}

Motivation: We have acknowledged the write to the host, but not destaged it. Upon the restore, we will lose the data. If the next split is a reverse split, we're in trouble.
Microcode: verify_parms_suborder_12, after vps12_cmd_ok

### 4. BFM(|) is invalid, if revsplitcheck bit is on {R}

Motivation: If the next split is a reverse split, we do not have the source for the copy.
Microcode: get_invalid_counts_establish, called from verify_parms_suborder_12
Microcode: verify_parms_suborder_12, after vps12_cmd_ok

## *B. Write Pendings*

### 1. If MMb is WP, carry over cache slot to STD

Motivation: To propagate the write pending to the standard.
Microcode: pyramid: hdc_establish, at skip_reset_wp; cache: update_invalid_bits_for_bcv_device at uib_r_cyls_loop, called from hdc_estalish.

### 2. If BCV is WP, but MMb is not, delete WP, mark track not in cache and mark the slot invalid for BFM(&)

Motivation: The write pending will be propagated to the standard because of the invalidation, so we don't need to keep the WP indication; however, in case of a future reverse split, we need to mark the track invalid.
Microcode: pyramid: hdc_establish, at skip_reset_wp; cache: update_invalid_bits_for_bcv_device at uib_r_cyls_loop, called from hdc_estalish.

### 3. Delete all WP to STD, and mark all such as invalid for SCM(&)

Motivation: Establish flows can not handle write pendings; this is redundant because of the HA reject, done under lock of the device.
Microcode: reset_regular_dv_wp_on_restore (same as reset_secondary_wp), called from update_-invalid_bits_for_bcv_device, called from hdc_establish.

## *C. Insure Purpose*

### 1. If m4b is invalid, mark SCM(&) invalid

Motivation: So a write to the BCV will be propagated to the standard.
Microcode: update_invalid_bits_for_bcv_device, called from hdc_establish.

### 2. If MMs is invalid, mark SCM(&) invalid

Motivation: So a write to the standard will be overwritten by the corresponding track on the BCV.
Microcode: update_invalid_bits_for_bcv_device, called from hdc_establish.

### 3. Set MMs to valid

Motivation: So the source of the copy exists.
Microcode: update_invalid_bits_for_bcv_device, called from hdc_establish.

## *D. Continuity/Persistence/Recovery*

### 1. If BFM(|) is invalid, mark track as changed for DDF {D}

Motivation: Upon an IML, we would lose the BFM invalid mark, which probably indicates the data hasn't been synced up completely yet. In case the next split is differential, we need to remember this.
Microcode: update_invalid_bits_for_bcv_device, called from hdc_establish.

| cdt | Split |
|---|---|

*A. HA Reject Conditions*

### 1. SCM(&) invalid, unless forced

Motivation: The standard would be left with a track wholly invalid.
Microcode: get_invalid_counts_terminate, called from verify_parms_suborder_13

### 2. MMs invalid, unless forced

Motivation: The BCV would end up with a track completely invalid.
Microcode: get_invalid_counts_terminate, called from verify_parms_suborder_13

### 3. WP limit

Motivation: In case we don't have space to duplicate the cache slots.
Microcode: get_and_check_regular_device, called from verify_parms_suborder_13

*B. Write Pendings*

### 1. If SCM(|) and MMs are WP, duplicate cache; watch out for WP limit

Microcode: copy_and_update_dv_tables, at case 2.

### 2. If MMs is WP, carry over cache slot to BCV

Microcode: copy_and_update_dv_tables, at case 1.

### 3. [If SCM(|) is WP, leave it be]

Motivation: To carry a write pending to its destination.
Microcode: copy_and_update_dv_tables, at case 3.

*C. Insure Purpose*

### 1. Set BFM(&) to invalid

Motivation: To initiate a copy to the BCV fixed mirrors from the just split moving mirror.
Microcode: compute_invalid_flags_on_split, called from copy_and_update_dv_tables.

### 2. Set MMb to valid

Motivation: It might be that a previous operation left MMb invalid; since the data comes from another device, we can and should forget about the previous invalid state of the MMb.
Microcode: invalid flags are zeroed early in the copy_and_update_dv_tables loop.

*D. Continuity/Persistence/Recovery*

### 1. Mark m4b valid

Motivation: Reset the logging mirror position so we can start keeping track of the writes to the BCV for the next attachment operation.
Microcode: compute_invalid_flags_on_split, called from copy_and_update_dv_tables.

## 2. Mark all tracks as unchanged for DDF and delete DDF session

Motivation: This is the standard way to get rid of the session.
Microcode: manage differential split, called from hdc_terminate.

## 3. [If MMs is invalid, leave it be]

Motivation: If we forced a split, we use this to remember the unsynced tracks.
Microcode: compute_invalid_flags_on_split, called from copy_and_update_dv_tables.

## 4. If MMs is invalid, set BCV(&) invalid

Motivation: Carry over the invalid tracks in case of a force.
Microcode: compute_invalid_flags_on_split, called from copy_and_update_dv_tables.

# ddt Differential Split {D}

## *A. HA Reject Conditions*

### 1. SCM(&) invalid, unless forced

Motivation: The standard would be left with a track wholly invalid.
Microcode: get_invalid_counts_terminate, called from verify_parms_suborder_13

### 2. MMs invalid, unless forced

Motivation: The BCV would end up with a track completely invalid.
Microcode: get_invalid_counts_terminate, called from verify_parms_suborder_13

### 3. WP limit

Motivation: In case we don't have space to duplicate the cache slots.
Microcode: get_and_check_regular_device, called from verify_parms_suborder_13

## *B. Write Pendings*

### 1. If SCM(|) and MMs are WP, duplicate cache; watch out for WP limit

Microcode: copy_and_update_dv_tables, at case 2.

### 2. If MMs is WP, carry over cache slot to BCV

Microcode: copy_and_update_dv_tables, at case 1.

### 3. [If SCM(|) is WP, leave it be]

Motivation: To carry a write pending to its destination.
Microcode: copy_and_update_dv_tables, at case 3.

## *C. Insure Purpose*

### 1. If DDF is marked changed, set BFM(&) invalid

Motivation: To insure all changed tracks (and only them) are copied to the fixed mirrors of the BCV.
Microcode: compute_invalid_flags_on_split, called from copy_and_update_dv_tables.

### 2. Set MMb to valid

Motivation: It might be that a previous operation left MMb invalid; since the data comes from another device, we can and should forget about the previous invalid state of the MMb.
Microcode: invalid flags are zeroed early in the copy_and_update_dv_tables loop.

## *D. Continuity/Persistence/Recovery*

### 1. Mark m4b valid

Motivation: Reset the logging mirror position so we can start keeping track of the writes to the BCV for the next attachment operation.
Microcode: compute_invalid_flags_on_split, called from copy_and_update_dv_tables.

## 2. Mark all tracks as unchanged for DDF

Motivation: So we can start logging writes to the standard again.
Microcode: compute_invalid_flags_on_split, called from copy_and_update_dv_tables.

## 3. [If MMs is invalid, leave it be]

Motivation: If we forced a split, we use this to remember the unsynced tracks.
Microcode: compute_invalid_flags_on_split, called from copy_and_update_dv_tables.

## 4. If MMs in invalid, set BCV(&) invalid

Motivation: Carry over the invalid tracks in case of a force.
Microcode: compute_invalid_flags_on_split, called from copy_and_update_dv_tables.

| crt | Reverse Split {R} |
|---|---|

*A. HA Reject Conditions*

### 1. SCM(&) invalid

Motivation: The standard would be left with a track wholly invalid.
Microcode: get_invalid_counts_terminate, called from verify_parms_suborder_13

### 2. BFM(&) invalid

Motivation: The BCV would end up with a track completely invalid, which could not be recovered.
Microcode: get_invalid_counts_terminate, called from verify_parms_suborder_13

*B. Write Pendings*

### 1. If SCM(|) and MMs are WP, delete MMs WP

Motivation: Since the track will be copied over from the BCV fixed mirrors anyway, there's no need to carry the WP.
Microcode: copy_and_update_dv_tables, at case 2.

### 2. If MMs is WP, delete WP and mark track not in cache

Motivation: Since the track will be copied over from the BCV fixed mirrors anyway, there's no need to carry the WP.
Microcode: copy_and_update_dv_tables, at case 1.

### 3. [If SCM(|) is WP, leave it be]

Motivation: Standard is untouched by operation.
Microcode: copy_and_update_dv_tables, at case 3.

*C. Insure Purpose*

### 1. Set MMb to invalid

Motivation: To initiate a copy to the BCV moving mirror from the fixed mirrors.
Microcode: compute_invalid_flags_on_split, called from copy_and_update_dv_tables.

*D. Continuity/Persistence/Recovery*

### 1. Set m4b to invalid

Motivation: In case of a future incremental restore of re-establish, we need to keep track of the fact that the BCV is (possibly) completely out-of-sync with the standard.
Microcode: compute_invalid_flags_on_split, called from copy_and_update_dv_tables.

### 2. Mark all tracks as unchanged for DDF and delete DDF session

Motivation: This is the standard way to get rid of the session.
Microcode: manage differential split, called from hdc_terminate.

## 3. [If BFM(|) is invalid, leave it invalid]

Motivation: In case an establish was done before the previous sync was finished.
Microcode: compute_invalid_flags_on_split, called from copy_and_update_dv_tables.

| drt | Reverse Differential Split {D, R} |
|---|---|

*A. HA Reject Conditions*

### 1. SCM(&) invalid.

Motivation: The standard would be left with a track wholly invalid.
Microcode: get_invalid_counts_terminate, called from verify_parms_suborder_13

### 2. BFM(&) invalid.

Motivation: The BCV would end up with a track completely invalid. [This will never get fixed!]
Microcode: get_invalid_counts_terminate, called from verify_parms_suborder_13

*B. Write Pendings*

### 1. If SCM(|) and MMs are WP, delete MMs WP.

Motivation: Since the track will be copied over from the BCV fixed mirrors anyway, there's no need to carry the WP.
Microcode: copy_and_update_dv_tables, at case 2.

### 2. If MMs is WP, delete WP and mark track not in cache.

Motivation: Since the track will be copied over from the BCV fixed mirrors anyway, there's no need to carry the WP.
Microcode: copy_and_update_dv_tables, at case 1.

### 3. [If SCM(|) is WP, leave it be.]

Motivation: Standard is untouched by operation.
Microcode: copy_and_update_dv_tables, at case 3.

*C. Insure Purpose*

### 1. If DDF is marked as changed, set MMb to invalid.

Motivation: To initiate a copy to the BCV moving mirror from the fixed mirrors, of only those tracks that changed.
Microcode: compute_invalid_flags_on_split, called from copy_and_update_dv_tables.

*D. Continuity/Persistence/Recovery*

### 1. [If BFM(|) is invalid, leave it invalid.]

Motivation: In case an establish was done before the previous sync was finished.
Microcode: compute_invalid_flags_on_split, called from copy_and_update_dv_tables.

### 2. If BFM(&) is invalid, set MMb invalid.

Motivation: If the source of the copy is invalid, we don't have the data.
Microcode: compute_invalid_flags_on_split, called from copy_and_update_dv_tables.

## 3. If MMb is invalid, set m4b to invalid.

Motivation: In case of a future incremental restore of re-establish, we need to keep track of the fact that the BCV track is (possibly) out-of-sync with the standard track.
Microcode: compute_invalid_flags_on_split, called from copy_and_update_dv_tables.

## 4. Mark all tracks as unchanged for DDF.

Motivation: So we can start logging writes to the standard again.
Microcode: compute_invalid_flags_on_split, called from copy_and_update_dv_tables.

# III. Checklists

For each operation we make certain we have not forgotten anything behind.

| cde | Establish |
|---|---|

| | *if WP* | *become WP* | *if invalid* | *becomes invalid* |
|---|---|---|---|---|
| **SCM** | - | - | HA reject | - |
| **MMs** | | | - | always |
| **DDF** | | | - | if BFM invalid |
| **MMb** | HA reject if rbit; make BFM invalid; delete WP | - | - | - |
| **BFM** | | - | HA reject if rbit; mark DDF changed | if BCV WP |
| **m4b** | | | - | - |

| dde | Re-Establish |
|---|---|

| | *if WP* | *become WP* | *if invalid* | *becomes invalid* |
|---|---|---|---|---|
| **SCM** | - | - | HA reject | - |
| **MMs** | | | - | if m4b invalid; if MMb invalid |
| **DDF** | | | - | if BFM invalid |
| **MMb** | HA reject if rbit; make BFM invalid; delete WP | - | mark MMs invalid | - |
| **BFM** | | - | HA reject if rbit; mark DDF changed | if BCV WP |
| **m4b** | | | make MMs invalid | - |

cre Restore

| | *if WP* | *become WP* | *if invalid* | *becomes invalid* |
|---|---|---|---|---|
| **SCM** | HA reject; delete WP | - | - | always |
| **MMs** | | | becomes valid | if MMb invalid |
| **DDF** | | | - | if BFM invalid |
| **MMb** | HA reject if rbit; make BFM invalid; delete and/or carry WP to STD | - | HA reject | - |
| **BFM** | | - | HA reject if rbit; mark DDF changed; make STD invalid | if BCV WP |
| **m4b** | | | - | - |

dre Incremental Restore

| | *if WP* | *become WP* | *if invalid* | *becomes invalid* |
|---|---|---|---|---|
| **SCM** | HA reject; mark SCM invalid; delete WP | - | - | if MMs invalid; if m4b invalid; if SCM WP |
| **MMs** | | | make SCM invalid; becomes valid | if MMb invalid |
| **DDF** | | | - | if BFM invalid |
| **MMb** | HA reject if rbit; make BFM invalid; delete and/or carry WP to STD | - | HA reject | - |
| **BFM** | | - | HA reject if rbit; mark DDF changed; make STD invalid | if BCV WP |
| **m4b** | | | make SCM invalid | - |

cdt Split

| | *if WP* | *become WP* | *if invalid* | *becomes invalid* |
|---|---|---|---|---|
| **SCM** | delete and/or carry WP to BCV; see rules. | - | HA reject, unless forced | - |
| **MMs** | | - | HA reject, unless forced; mark MMb invalid | - |
| **DDF** | | | Mark all unchanged; delete session | |
| **MMb** | | if MMs is WP | becomes valid | if MMs invalid |
| **BFM** | | | - | always |
| **m4b** | | | becomes valid | - |

ddt Differential Split

| | *if WP* | *become WP* | *if invalid* | *becomes invalid* |
|---|---|---|---|---|
| **SCM** | delete and/or carry WP to BCV; see rules. | - | HA reject, unless forced | - |
| **MMs** | | - | HA reject, unless forced; mark MMb invalid | - |
| **DDF** | | | Mark all unchanged | |
| **MMb** | | if MMs is WP | becomes valid | if MMs invalid |
| **BFM** | | | - | if DDF changed; if MMs invalid |
| **m4b** | | | becomes valid | - |

## crt Reverse Split

| | *if WP* | *become WP* | *if invalid* | *becomes invalid* |
|---|---|---|---|---|
| **SCM** | delete or carry WP to BCV; see rules. | - | HA reject, unless forced | - |
| **MMs** | | - | - | - |
| **DDF** | | | Mark all unchanged; delete session | |
| **MMb** | | - | - | always |
| **BFM** | | | HA reject, unless forced | - |
| **m4b** | | | - | always |

## drt Reverse Differential Split

| | *if WP* | *become WP* | *if invalid* | *becomes invalid* |
|---|---|---|---|---|
| **SCM** | delete or carry WP to BCV; see rules. | - | HA reject, unless forced | - |
| **MMs** | | - | - | - |
| **DDF** | | | Mark all unchanged | |
| **MMb** | | - | - | if DDF changed; if BFM invalid |
| **BFM** | | | HA reject, unless forced; set MMb invalid | - |
| **m4b** | | | - | if MMb invalid |

# IV. Proof

We prove a number of facts related to the correct functioning of TimeFinder.

A. Assume the STD holds dataset A and the BCV holds B. In the absence of IO (write-pending tracks), an establish operation *x* followed by a terminate operation *y* will affect the data in the following way:

| *x* | *y* | STD | BCV |
|---|---|---|---|
| *de | *dt | A | A |
| *de | *rt | A | B |
| *re | *dt | B | B |
| *re | *rt | B | B |

PROOF: ...

B. Influence of IML...

PROOF: ...

C. Any write acknowledged to the host will be accounted for...

PROOF: ...

EXHIBIT 3

LAW OFFICES OF
GEORGE A. HERBSTER

INTELLECTUAL PROPERTY AND RELATED CAUSES

SUITE 303, HARBOR'S POINT
40 BEACH STREET
MANCHESTER, MASSACHUSETTS 01944

TELEPHONE: (978) 526-8111

FACSIMILE: (978) 526-8062
EMAIL: herbster@star.net

December 1, 1998

Mr. Mattieu Gagne
EMC Corporation
171 South Street
Hopkinton, MA 01748

Re: Patent Application For Far Point Cascading
EMC File: 98-096
My File: E30-029

Dear Mattieu,

Enclosed are 5 sheets of drawings and some claims for the far-point cascading patent application.

The three sheets marked 12/19 through 14/19 are drawings from the original BCV application. They show the operation that was presented to the United States Patent and Trademark Office for the "split" and "reestablish" commands. The other two sheets depict my understanding of one embodiment with three different locations. The other depicts my concept of the "differential split" implementation.

The attached claims are drafted to define the crux of the invention. Claims 1 and 6 provide two different approaches. Claims 2 through 5 are "dependent" claims that add steps to the invention as set forth in claim 1.

Please call me after you have a chance to review this material.

Sincerely,

George A. Herbster

GAH:mh

Enclosures
cc: J. Gunther, Esq.